TITLE:

高階論理ユニフィケーションを用いた知識処理(計算アルゴリズムと計算量の基礎理論)

AUTHOR(S):

原尾, 政輝; 岩沼, 宏治; 安孫子, 力雄

CITATION:

原尾, 政輝 ...[et al]. 高階論理ユニフィケーションを用いた知識処理(計算アルゴリズムと計算量の基礎理論). 数理解析研究所講究録 1989, 695: 98-107

ISSUE DATE:

1989-06

URL:

http://hdl.handle.net/2433/101400

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
695巻 1989年 98-107

高階論理ユニフィケーションを用いた知識処理

山形大工情報工　原尾　政輝 (Masateru HARAO)
山形大工情報工　岩沼　宏治 (Koji IWANUMA)
山形大工情報工　安孫子力雄 (Rikio ABIKO)

1. はじめに

一階述語論理に基づく知識表現と推論機構は理論的に明快で汎用性を持っているが，その反面，構造を持った知識表現やメタ推論と言った性質の取り扱いが複雑になる．その欠点を補うための一手法として，関数変数や述語変数を許す高階論理へ拡張することが考えられる．一般に高階論理は型付きλ論理にいくつかの論理的公理と推論規則を加えて構成される．よく知られているように，型付きλ式は，正則性といった本質的な条件を満たし，かつ，Huet[5]により提案された有効なユニフィケーションアルゴリズムが存在する．

本稿では，型付きλ論理を用いた高階言語を定義し，Huetのアルゴリズムを基に，ユニフィケーションが可解となるクラスの例を示す．また，実装したユニフィケーションプログラムの概要について述べ，再帰方程式のスキーマ認識を題材にメタ推論などの知識処理への適応性を示す．

2. 型付きλ式と高階論理

2.1 型付きλ論理

型付きλ論理に基づいた高階言語を定義する．

【定義1】基本となる型の有限集合を T_0 とするとき，型の集合Tとは $T_0 \subseteq T$ かつ以下の条件を満たす最小の集合である．

$\alpha, \beta \in T$, ならば $(\alpha \to \beta) \in T$ □

以下では $o, i \in T_0$ であり，o, iはそれぞれ真理値の型，個体領域の型とする．

【定義2】型 α の階数 $O(\alpha)$ を次にように定義する．

$O(\alpha) = 1$ if $\alpha \in T_0$

$O((\alpha \to \beta)) = \max\{O(\alpha)+1, O(\beta)\}$ □

各型 α に対して変数の可算無限集合と特別な定数記号として型 $(o \to o)$ を持つ $\neg$，型 $(o \to (o \to o))$ を持つ $\vee$，各型 α に対し型 $((\alpha \to o) \to o)$ を持つ Π と型 $((\alpha \to o) \to \alpha)$ を持つ ι を含む定数記号の可算集合を考え，その全体集合をそれぞれV，Cとする．集合 $V \cup C$ の元を原子記号と呼ぶ．以下，変数に対して $x, y, z, \ldots$ を用いる．

【定義3】項の集合Terと項tの型 $\tau(t)$ を定義する．

1) $t \in V \cup C$ ならば $t \in Ter$

2) $s, t \in Ter$, $\tau(s) = (\alpha \to \beta)$, $\tau(t) = \alpha$ ならば

$(s\ t) \in Ter$, $\tau((s\ t)) = \beta$

3) $t \in Ter$ ならば $\lambda x \cdot t \in Ter$ $\tau(\lambda x \cdot t) = (\tau(x) \to \tau(t))$ □

束縛変数，自由変数については通常の通り定義する．

【定義4】項tがk階であるとは，tに出現する自由変数が高々k階である場合を言う． □

以下，項をs, t, ... と略記する．

【定義5】型oを持つ項を論理式という． □

以下，論理式をA，B，．．．と略記し，$((\vee A)\ B)$ を $A \vee B$，$(\neg A) \vee B$ を $A \supset B$，$\Pi(\lambda x \cdot A)$ を $\forall x \cdot A$，$\forall P \cdot (P(s) \supset P(t))$ を $s = t$ と略記する．他の論理記号についても通常のように定義する．

【定義6】２つの項s,tがλ変換可能であることをs⇔ｔで表わす．以下，t[x/s]は t∈Ter でのxの自由な出現を s∈Ter で置き換える操作である．

（α変換）ｙが ｔに出現しなければ，

λ x.t ⇔ λ y.(t[x/y]) 但し，$\tau(y)=\tau(x)$

（β変換）s中に出現する自由変数とtの束縛変数が違うならば，

((λ x.t) s) ⇔t[x/s] 但し，$\tau(s)=\tau(x)$ □

2.2 形式体系

公理は一階論理の公理に加えて，次のものを仮定する．

a) A ≡ B ⊃ A = B

b) ∀ x･((s x) = (t x)) ⊃ s = t

但し，$\tau(s)=\tau(t)=(\alpha\rightarrow\beta)$，$\tau(x)=\alpha$

c) (s x) ⊃ (s (ι s))

但し，$\tau(s)=(\alpha\rightarrow o)$ $\tau(x)=\alpha$

推論規則は通常のModus ponens, Universal generalization,代入のほか，上記のλ変換を用いる．

証明，定理は通常のように定義し，Ａが定理であることを ⊢Ａ と略記する．

2.3 意味論

【定義7】フレームＦとは以下の条件を満たす領域D_αの族$\{D_\alpha\}_{\alpha\in T}$である．

1) $D_o=\{true,false\}$

2) $D_{(\beta\rightarrow\alpha)}$は $[D_\beta\rightarrow D_\alpha]$ の空でない部分集合． □

【定義8】解釈Ｍとは以下の条件をみたす２項組$\langle\{D_\alpha\}_\alpha,m\rangle$である．$\{D_\alpha\}_\alpha$はフレーム，ｍは定数記号の解釈を決める意味関数で以下の条件を満たす．

m(¬),m(∨)はD_o上の通常の論理否定関数と論理和関数である．また，m(Π)は$D_{(\alpha\to o)}$の元fがD_αのすべての元を$true\in D_o$に射影するときfにtrueを割り当て，そうでなければ，falseを割り当てる関数．m(ι)は$D_{(\alpha\to o)}$の元fがD_αの唯一の元dに対し$true\in D_o$を割り当てるならば，fに対しそのdを割り当てる関数．その他の定数記号に対して，mはD_αの適当な元を割り当てる関数． □

【定義9】Fをフレーム$\{D_\alpha\}_\alpha$とするとき，F上の割当ψとは各変数x_αに対して領域D_α上の元を割り当てる関数である．また，割当$\psi[x_\alpha/d]$とは変数x_αに$d\in D_\alpha$を割り当て，それ以外の変数にはψと同じ元を割り当てる割当である． □

【定義10】解釈$M=<\{D_\alpha\}_\alpha,m>$が一般モデルであるとは，各型$\alpha$に対し領域$D_{(\alpha\to(\alpha\to o))}$が$D_\alpha$上の恒等関係を含み，かつ，以下の条件を満たす2項関係Vが存在するときを言う．

1) M上の任意の割り当てψと型αの項sに対して，

$$V_\psi(s_\alpha)\in D_\alpha$$

2) M上の任意に割当ψと任意の変数$x,\tau(x)=\alpha$と項s，t，$\tau(s)=\beta$，$\tau(t)=(\beta\to\gamma)$に対して

a) $V_\psi(x)=\psi(x)$

b) $V_\psi(s)=m(s)$ if $s\in Ter$

c) $V_\psi((t\ s))=V_\psi(t)V_\psi(s)$

d) $V_\psi(\lambda x\cdot s)=$Dの元dに対し$V_{\psi[x_\alpha/d]}(s)$を返す

D_αからD_βへの関数 □

【定義11】論理式Aが一般モデルMで妥当であるとは，M上の任意の割当ψに対し，$V_\psi(A)=true$である時を言い，Aが全ての一般モデル上で妥当であるとき$\models A$と略記する． □

【定理】(Henkinの完全性定理) Aを論理式とするとき，以下が成り立つ．

$$\vdash A \quad \text{iff} \quad \models A$$ □

3. ユニフィケーションアルゴリズム

3.1 ユニフィケーション

以下, $\alpha_1,\ldots,\alpha_n \in T, \beta \in T_0$ であるとき, 型$(\alpha_1 \to (\alpha_2 \to (\ldots(\alpha_n \to \beta)\ldots))$を$(\alpha_1,\ldots,\alpha_n \to \beta)$と略記する. また, 項$(\ldots((s\ t_1)\ t_2)\ldots t_p)$を $s(t_1,t_2,\ldots,t_p)$と略記し, $\lambda x_1 \ldots \lambda x_n \cdot s$を $\lambda x_1 \ldots x_n \cdot s$と略記する. 但し, $x_1,\ldots,x_n$は相異なる変数である.

sが$\tau(s)=(\alpha \to \beta)$の項とするとき $\vdash s = \lambda x \cdot s(x)$ が一般に成り立つことから次の変換が定義できる.

【定義12】(η 変換) $\tau(s)=(\alpha \to \beta)$のとき

$$s \Leftrightarrow \lambda x \cdot s(x) \qquad \square$$

【定義13】β, η 変換において左から右への変換が適用できない項を正規形という. $\square$

正規形 t は, $t = \lambda x_1 \ldots x_n \cdot s(t_1,\ldots,t_p)$の形(部分式 t_i も同じ形)を取る. ここで, s を項 t のヘッドと呼ぶ. ヘッド sが, $s \in C \cup \{x_1,\ldots x_n\}$であるとき, tをrigid項と呼び, 自由変数であるとき flex項と呼ぶ.

【定義14】代入 σ とは以下の形をした有限集合である.

$$\sigma = \{x_1/t_1,\ldots,x_n/t_n\}$$

但し, $\tau(x_i)=\tau(t_i)$, t_iは正規形 $(1 \leqq i \leqq n)$

項tに対し代入 σ を施した項 σtを項$[\lambda x_i \ldots x_n \cdot t](t_1,\ldots,t_n)$の正規形と定義する. $\square$

【定義15】$\tau(t_1)=\tau(t_2)$なる正規項t_1,t_2のユニファイヤとは, $\sigma t_1 = \sigma t_2$ なる代入 σ のことを言う. $\square$

【定義16】 不一致集合は$\tau(t_1)=\tau(t_2)$なる正規形t_1,t_2の順序対$<t_1,t_2>$の有限集合である. $\square$

ユニフィケーションアルゴリズムはSIMPLとMATCH，それらを用いてマッチング木を構成するmainアルゴリズムからなる．

SIMPLは不一致集合Nを受け取り，N中のともにrigidである項の対に対し，共通なヘッドを取り除き，部分項の対に分解する．このとき，明らかにNがユニファイ可能であればSUCCESSを返し，不可能ならばFAILを返す．どちらとも分からなければ，新しく部分項の対を加えた不一致集合N'を返す．

MATCHは不一致集合N中の一方がflex項である対を受け取り，代入の有限集合を返す．

mainアルゴリズムは$\tau(t_1)=\tau(t_2)$なる項t_1，t_2に対して，マッチング木を以下のように構成する．

節点のラベルは，不一致集合NかSUCCESSまたはFAILである．

1) 木の根はN_0=SIMPL({<t_1,t_2>})とする．

2) もしNがSUCCESSかFAILでなければNの中の任意の対<t_1',t_2'>を取り出し，MATCHにより代入の集合{σ_1,...,σ_k}を求める．

3) SIMPL(σ_iN)を子節点としσ_iでラベル付した弧を張る．□

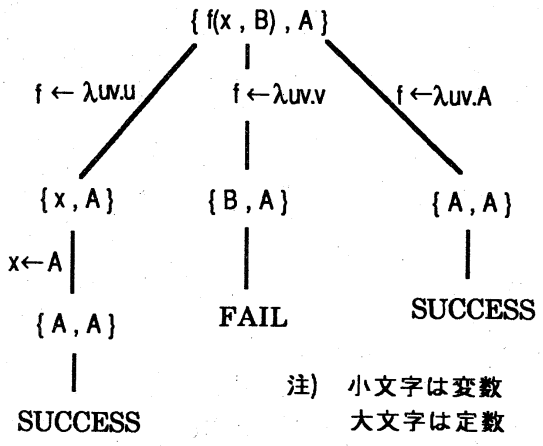

図1 matching tree

マッチング木上で根から SUCCESS に至るパスにラベル付された代入の合成は，t_1,t_2のユニファイヤになること（健全性）が証明されている．このマッチング木を幅優先で構成する手続きは独立なユニファイヤ全て（完全性）を，重複なし（非冗長）に数え上げられる．[5]

我々はすでにユニフィケーションアルゴリズムを計算機上に実装している．その具体例は付録に示す．

3.2 ユニフィケーション計算の可解なクラス

一般に，２階以上の型付きλ式のユニフィケーション問題は決定不可能であるが，可解となる部分問題もいくつか知られている．まず，自明なものとしては，高々２階かつ４階以上の定数が出現せず，さらに片方にしか自由変数が現われない２つの項間のユニフィケーション問題がある．この問題に対しては，上述の手続きの停止性が保証されている．このクラスに属する代表的なものとして後述するスキーマ認識問題がある．

【命題】 ２つのλ式をimitation 代入だけを使用してユニファイ可能か否かの判定は可解である． □

4. 再帰方程式のスキーマ認識・変換

対象とする知識データにある処理をする場合，個別の知識に対しそれぞれの処理法を記述していたのでは効率がよくない．そこで，ある共通の概念をメタ表現として表わし，その上でのメタ規則で処理を行う手法が考えられる．このメタ推論に対して前述の高階ユニフィケーションが有用である．本節では回路合成におけるスキーマ認識[9]への高階ユニフィケーションの応用について述べる．

4.1 再帰方程式と再帰スキーマ

【定義17】 再帰方程式は以下の形をした関数方程式である.

$f_1(x_1^1,\ldots,x_{n_1}^1) <= t_1$

.... 但し, $t_i(1 \leqq i \leqq p)$は再帰表現

$f_p(x_1^p,\ldots,x_{n_p}^p) <= t_p$ □

再帰方程式は, 型付きλ式を用いて次のように表せる.

$$\lambda u_1 \ldots u_n \cdot Y(\lambda f_1,\ldots,f_p \cdot <\lambda x_1^1 \ldots x_{n_1}^1 \cdot t_1, \ldots, \lambda x_1^p \ldots x_{n_p}^p \cdot t_p>, u_1,\ldots,u_n)$$

ここで, <>は直積関数, Yは不動点演算子であり, その式の文脈から型付できる定数記号である.

また, 関数変数を持つ再帰方程式を再帰スキーマと呼ぶ.

例1. 再帰スキーマの例 以下, 大文字を定数, 小文字を変数とする.

schema1: $f(x) <= \text{if } p(x) \text{ then } g(x) \text{ else } f(h_1(x))$.

schema2: $f(x) <= \text{if } p(x) \text{ then } g(x) \text{ else } h_2(f(g_1(x)))$.

schema3: $f(x) <= \text{if } p(x) \text{ then } g(x) \text{ else } h_3(g_1(x), f(g_2(x)))$.

$\tau(f)=\tau(g)=\tau(g_1)=\tau(g_2)=\tau(h_1)=\tau(h_2)=(i \to i)$

$\tau(h_3)=(i, i \to i), \tau(p)=(i \to o), \tau(x)=i$

4.2 再帰方程式の認識と変換

再帰スキーマSがS'に変換可能であると仮定した場合, 再帰方程式eに対してe = σSなる代入σが存在すればe' = σS'で得られるe'はeと等価な再帰方程式であることが分かる. これは1つのメタ推論である.

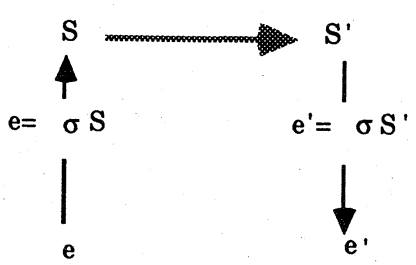

図2 スキーマ認識

例２．リストを反転する関数 rev は 上のschema3と認識され，ユニファイヤは以下のσ∪σ$_1$，σ∪σ$_2$，σ∪σ$_3$の３つが作られる．

```
rev(x)<= if Null(x) then Nil
         else Append(rev(Cdr(x)),Cons(Car(x),Nil)).
```

σ ={p/λ u.Null(u),g/λ u.Nil, g$_2$/λ u.Cdr(x)}

σ$_1$={h$_3$/λ xy.Append(y,x),g$_1$/λ u.Cons(Car(u),Nil)}

σ$_2$={h$_3$/λ xy.Append(y,Cons(x,Nil)),g$_1$/λ u.Car(u)}

σ$_3$={h$_3$/λ xy.Append(y,Cons(Car(x),Nil)),g$_1$/λ u.u}

従って，schema3が別のスキーマに等価変換されるとき上のrevは，この３つのユニファイヤにしたがって，三つの等価な方程式に変換可能となる．

５．ユニフィケーションアルゴリズムの実装について

［構文］入力λ式の構文は下記のようなものである．項は型付きλ式の定義を満たすように組み立てられるものとする．

```
<項>      ::=  <適用>|<λ抽象>|<アトム>
<適用>    ::=  (<項>,<項列>)
<λ抽象>   ::=  <変数>!<項>
<アトム>  ::=  <変数>|<定数>
<項列>    ::=  <項>,<項列>|<項>
<変数>    ::=  @<英字列>
<定数>    ::=  <英字列>
```

但し，英字列は英小文字ではじまる文字列とする．

入力されたλ式は lambda([束縛変数],[ヘッド|引数])なる内部表現に変換される．

入力される項の変数及び定数の型は最初に定義する．また，ユニフィケーションの過程でつくられる新しい変数については，対応する型を自動的につける．

6. むすび

本稿では高階論理言語の定義を与え，それを用いてメタ推論を実現するためのユニフィケーションアルゴリズムの実装例を示し，スキーマ認識へのを応用を述べた．高速にメタ推論を行なわせるには，その中心となるユニフィケーションになんらかの計算戦略（または，ヒューリスティック）をいれる必要がある．また，スキーマ認識に関しては，スキーマが変換可能となる条件がユニファイした再帰方程式に対して成り立つかどうか，構文情報からだけでは判断できないものもあり，今後の研究課題である．

<<参考文献>>

[1] P.B.Andrews: Resolution in Type Theory,JSL,vol.36, pp414-432,1971

[2] A.Church: A formulation of the simple theory of types,JSL,vol5,pp56-68,1940

[3] L.Henkin: Completeness in the theory of types, JSL,vol.15,pp81-91,1966

[4] G.P.Huet: A Mechanization of Type Theory,proc.of 3rd IJCAI,1973

[5] G.P.Huet: A Unification algorithm for typed λ-calculas,TCS vol.1,pp27-57,1975

[6] G.Huet,B.Lang: Proving and Applying Program Transformations Expressed with Second-Order Patterns,Acta Informatica vol.11,pp31-55,1978

[7] D.A.Miller,G.Nadathur: Higher-Order Logic Programming,LNCS.225,pp449-462,1986

[8] 岩沼,原尾：高階論理におけるユニフィケーションについて,信学技報,COMP87-27,1987

[9] 原尾：時空間様相論理に基づくシステム記述言語の設計とその回路自動設計問題への応用,科研費研究報告書(一般(C)60580016),1987-3